µC/USB-Bulk

USB-Bulk-Driver for
Embedded Aplications

Version 1.0

Manual Rev. 0

Micriμm Technologies Corporation

www.micrium.com
Empowering Embedded Systems

Disclaimer

Specifications written in this manual are believed to be accurate, but are not guaranteed to be entirely free of error. Specifications in this manual may be changed for functional or performance improvements without notice. Please make sure your manual is the latest edition. While the information herein is assumed to be accurate, Micrium Technologies Corporation (the distributor) assumes no responsibility for any errors or omissions and makes no warranties. The distributor specifically disclaims any implied warranty of fitness for a particular purpose.

Copyright notice

The latest version of this manual is available as PDF file in the download area of our website at www.micrium.com. You are welcome to copy and distribute the file as well as the printed version. You may not extract portions of this manual or modify the PDF file in any way without the prior written permission of Micrium Technologie Corporation. The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such a license.

© 2004 Micrium Technologies Corporation, Weston, Florida 33327-1848, U.S.A.

Trademarks

Names mentioned in this manual may be trademarks of their respective companies.
Brand and product names are trademarks or registered trademarks of their respective holders.

Registration

Please register the software via email. This way we can make sure you will receive updates or notifications of updates as soon as they become available.
For registration please provide the following information:

- Your full name and the name of your supervisor
- Your company name
- Your job title
- Your email address and telephone number
- Company name and address
- Your company's main phone number
- Your company's web site address
- Name and version of the product

Please send this information to: licensing@micrium.com

Contact address

Micrium Technologies Corporation
949 Crestview Circle
Weston, FL 33327-1848
U.S.A.
Phone : +1 954 217 2036
FAX : +1 954 217 2037
WEB : www.micrium.com
Email : support@micrium.com

Manual versions

This manual describes the software version 1.00. If any error occurs, please inform us and we will try to help you as soon as possible.
For further information on topics or routines not yet specified, please contact us.

Print date: 9/3/04

Manual version	Date	By	Explanation
1.0	040831	KG	Initial Version

Software versions

Software version	Date	By	Explanation
1.0	040831	RS	Initial Version

© 2004 Micrium Technologies Corporation

© 2004 Micrium Technologies Corporation

Table of Contents

1 Introduction 7
- 1.1 What is included 7
- 1.2 Purpose of this document 8
- 1.3 Assumptions 8
- 1.4 Requirements 8
- 1.5 Features 9
- 1.6 Typographic conventions for syntax 9
- 1.7 Data types 9

2 USB basics 11
- 2.1 Endpoint types 12
- 2.2 Setup phase / enumeration 13
- 2.3 Product / vendor Ids 14

3 Getting the target up 15
- 3.1 File structure 16
- 3.2 Start application 17
- 3.3 Configuration 18
- 3.4 Hardware dependent part 19
- 3.5 OS dependent part 23
- 3.6 Generic bulk stack 24

4 The Kernel mode driver (PC) 25
- 4.1 General info about the driver 26
- 4.2 Installing the driver 27
- 4.3 Recompiling the driver 31
- 4.4 The .inf file 32

5 Using the sample application 35
- 5.1 Running the sample application 36
- 5.2 Compiling the PC sample application 37

6 Configuration 39
- 6.1 Configuration files 40
- 6.2 USB_Conf.h 40
- 6.3 USB_Descriptors.c 42

7 Support 43
- 7.1 Problems with tool chain (compiler, linker) 44
- 7.2 Problems with hardware/driver 44
- 7.3 Contacting support 45
- 7.4 FAQ's 45

 © 2004 Micrium Technologies Corporation

8 Index 47

 © 2004 Micrium Technologies Corporation

Chapter 1

Introduction

The USB-Bulk stack has been designed to work on any embedded system with USB client controller. A hardware abstraction layer makes it possible to easily write or modify existing hardware access routines. It can be used with USB 1.1. or USB 2.0 devices.
The stack consists of an embedded side, which is shipped in source-code form, and a driver for the PC, which is typically shipped as executable (sys), but is also available as source code.
The purpose of the USB stack is to allow developers to quickly and painlessly develop software for an embedded device to communicate with a PC via USB. The communication is like a single, high speed, reliable channel (very similar to a TCP connection). It basically allows the PC to send data to the embedded target, the embedded target to receive these bytes and reply with any number of bytes. The PC is the USB host, the target is the USB client. The USB standard defines 4 types of communication: Control, isochronous, interrupt and bulk. Experience shows, that for most embedded devices bulk mode is the communication mode of choice. It allows usage of the full bandwidth of the USB bus.

Speed
The highest possible transfer rate on USB 1.1 (12 Mbit/sec) devices is app. 1.1 MByte per second. This data rate can indeed be achieved on fast system, such as ARM7.

1.1 What is included

The following table shows the contents of all USB bulk stack subdirectories:

Directory	Contents
Doc	Contains the USB-Bulk stack documentation
Driver	Contains the kernel mode USB driver for the PC (Win32, NT platform), the compiled driver (.sys). the .inf file required for installation, as well as the source code and a project for MSVC 6.0
SampleApp	Contains a PC sample project to help bring up and test the system
USBBulkStack	Contains the USB driver for the target hardware

 © 2004 Micrium Technologies Corporation

1.2 Purpose of this document

This guide describes how to install, configure and use the µC/USB Bulk stack for embedded applications. It also explains the internal structure of the software.

1.3 Assumptions

This guide assumes that you already possess a solid knowledge of the "C" programming language. If you feel that your knowledge of "C" is not sufficient, we recommend *The "C" Programming Language* by Kernighan and Richie, which describes the programming standard and, in newer editions, also covers the ANSI "C" standard. Knowledge of assembly programming is not required.

1.4 Requirements

The final purpose is usually to be able to run the software on a target system.

1.4.1 Target system (hardware)

Your target system must have a build in USB client.
The memory requirements are app. 6kB ROM and app. 1kb of RAM (used for buffering ... Systems with less RAM can also be used)

1.4.2 Development environment (compiler)

The CPU used is of no importance; only an ANSI-compliant "C" compiler is required. If your compiler has some limitations, please let us know and we will inform you if these will be a problem when compiling the software. Any compiler for 16/32/64-bit CPUs or DSPs that we know of can be used; most 8-bit compilers can be used as well. A C++ compiler is not required, but can be used. The application program can therefore also be programmed in C++ if desired.

 © 2004 Micrium Technologies Corporation

1.5 Features

- High speed
- The stack can be used with or without a RTOS.
- Start / test application supplied

1.6 Typographic conventions for syntax

This manual uses the following typographic conventions:

Style	Used for
Body	Body text.
`Keyword`	Text that you enter at the command-prompt or that appears on the display (i.e. system functions, file- or pathnames).
`Parameter`	Parameters in API functions.
`Sample`	Sample code in program examples.
`New Sample`	Sample code that has been added to an existing program example.
Warning	Important cautions or reminders.

1.7 Data types

Since "C" does not provide data types of fixed lengths which are identical on all platforms, the stack uses, in most cases, its own data types as shown in the table below:

Data type	Definition	Explanation
USB_I8	signed char	8-bit signed value
USB_U8	unsigned char	16-bit unsigned value
USB_I16	signed short	16-bit signed value
USB_U16	unsigned short	16-bit unsigned value
USB_I32	signed long	32-bit signed value
USB_U32	unsigned long	32-bit unsigned value

For most 16/32-bit controllers, the "C" data types used will work fine. However, the data types and can be configured in the main configuration file.

© 2004 Micrium Technologies Corporation

© 2004 Micrium Technologies Corporation

Chapter 2

USB basics

Before attempting to use the USB stack on your target, you should know some basics of USB. This chapter tries to explain the fundamentals and contains links for more information.

© 2004 Micrium Technologies Corporation

2.1 Endpoint types

The USB standard defines 4 types of communication: Control, isochronous, interrupt and bulk. Control transfers are used in the setup phase. The application can basically select one of the other 3 transfer types. For most embedded applications, bulk is the best choice since it allows the highest possible data rates.

Control transfers

Typically used to configure a device. It can also be used device specific purposes. Used to configure a device at attach time and can be used for other device-specific purposes, including control of other pipes on the device.

Isochronous transfers

Typically used for applications which need guaranteed speed. Isochronous transfer is fast, but with possible data loss. A typical use is audio, which requires a constant data rate.

Interrupt transfers

Typically used by devices that need guaranteed quick responses (bounded latency).

Bulk transfers

Typically used by devices that generate or consume data in relatively large and bursty quantities. Bulk transfer have wide dynamic latitude in transmission constraints. It can use all remaining available bandwidth, but with no guarantees on bandwidth or latency. Since the USB bus is normally not very busy, there is typically 90% or more of the bandwith available for USB transfers.

 © 2004 Micrium Technologies Corporation

2.2 Setup phase / enumeration

Before the target can communicate with the host, the host needs to get information from the target. This information is gathered in the initial setup phase. The information is contained in the descriptors, which are in the configurable section of the stack.
The most important part for identification of the target device are the product Id and vendor Id.
During the setup phase, the host also assign an address to the client. This part of the setup is called enumeration.

© 2004 Micrium Technologies Corporation

2.3 Product / vendor Ids

A vendor Id can be obtained from the USB Implementers Forum, Inc. (www.usb.org). This is necessary only when you finish development of the product; during the development phase, you can use the vendor and product Ids supplied as sample.

 © 2004 Micrium Technologies Corporation

Chapter 3

Getting the target up

The first step in getting the USB stack up and running is typically to compile it for your target system and to run it in the target system. This chapter explains how to do this.

© 2004 Micrium Technologies Corporation

3.1 File structure

The following table shows the contents of the Firmware subdirectories:

Directory	Contents
Application	Contains the application program. This is per default a modified echo server, which can be used to test the stack.
Config	Configuration file
System\CPU	Hardware interface routines for CPUs which have been tested so far.
System\OS	Operating systems dependent files which allow you to run the stack with different RTOSes or standalone.
USBBulk	Contains the actual USB stack data. Do not change the source code in this directory.

 © 2004 Micrium Technologies Corporation

3.2 Start application

The start application (in the application subfolder) is a modified echo server, which can be used to test the stack. The application receives data byte by byte, increments every single byte and sends it back to the host.

Source code of Start.c:

```
/*********************************************************
*             Micrium Technologies Corporation
*                Empowering embedded systems
*********************************************************
File    : Start.c
Purpose : Modified echo server to test the stack
--------- END-OF-HEADER ---------------------------------*/

#include "USB.h"

void MainTask(void) {
  //
  // Wait for configuration
  //
  while (USB_IsConfigured() == 0) {
    // HW_ToggleLED0(); // Toggle LED to indicate configuration
    USB_X_Delay(50);
  }
  //  HW_ClrLED0();

  //
  // Loop: Receive data byte by byte, send back (data + 1)
  //
  while (1) {
    char c;
    USB_Read(&c, 1);
    // SetLED(20);      // Flash LED to show receiption of data
    c++;
    USB_Write(&c, 1);
  }
}
```

The start application will of course later on be replaced by your real application program. For the purpose of getting the stack up and running as well as doing an initial test, the start application should not be modified, but instead compiled and linked with your project.

© 2004 Micrium Technologies Corporation

3.3 Configuration

The configuration section will later on be modified to match your real application. For the purpose of getting the stack up and running as well as doing an initial test, the configuration as shipped should not be modified.

© 2004 Micrium Technologies Corporation

3.4 Hardware dependent part

Routine	Explanation
USB_HW_Init()	Sets default values for EP0 and enables the various interrupts needed for USB operations.
USB_HW_FreeBuffer()	Let the USB controller know that received packet has been read and the buffer can now be re-used
USB_HW_memcpy	Memcopy (can be an accelerated form)
USB_HW_BufferAvailable()	Returns if a buffer is available.
USB_HW_SendData()	Sends out data (bulk mode) on EP1
USB_HW_Read()	Read the received data from the USB controller
USB_HW_Attach()	
USB_HW_UnloadEP0()	Called after a setup packet has been successfully evaluated. Tells the USB controller that the associated data buffer is no longer needed and can be freed.
USB_HW_StallEP0()	
USB_HW_SetAddress()	
USB_HW_EnableEP1()	
USB_HW_EP0_Send()	Sends a packet on EP0.
USB_HW_ClearOutPacketReady()	TBD
USB_HW_ClrSetupEnd	TBD

USB_HW_Init()

Description

Sets default values for EP0 and enables the various interrupts needed for USB operations.

Prototype

```
void USB_HW_Init(void);
```

USB_HW_FreeBuffer()

Description

Let the USB controller know that received packet has been read and the buffer can now be re-used

Prototype

```
void USB_HW_Init(void);
```


© 2004 Micrium Technologies Corporation

USB_HW_memcpy()

Description

Memcopy (can be an accelerated form)

Prototype

```
void USB_HW_memcpy(void * pDest, void * pSrc, unsigned Len);
```

USB_HW_BufferAvailable()

Description

Returns if a buffer is available.

Prototype

```
int USB_HW_BufferAvailable(void);
```

Return value

A 32-bit integer.

USB_HW_SendData()

Description

Sends out data (bulk mode) on EP1

Prototype

```
void void USB_HW_SendData(const void* pData, unsigned NumBytes);
```

USB_HW_Read()

Description

Returns a unique ID for the current task.

Prototype

```
void USB_HW_Read(void * pDest, int NumBytes, unsigned PacketOff);
```

USB_HW_Attach()

Description

Prototype

```
void USB_HW_Attach(void);
```

 © 2004 Micrium Technologies Corporation

USB_HW_UnloadEP0()

Description

Called after a setup packet has been successfully evaluated. Tells the USB controller that the associated data buffer is no longer needed and can be freed.

Prototype

```
void USB_HW_UnloadEP0(void);
```

USB_HW_StallEP0()

Description

Prototype

```
void USB_HW_StallEP0(void);
```

USB_HW_SetAddress()

Description

Prototype

```
void USB_HW_SetAddress(unsigned char Adr);
```

USB_HW_EnableEP1()

Description

Prototype

```
void USB_HW_EnableEP1(void);
```

void USB_HW_EP0_Send()

Description

Sends a packet on EP0

Prototype

```
void USB_HW_EP0_Send(const unsigned char * pData, unsigned char SendLength,
char SendNULLPacket)
```

 © 2004 Micrium Technologies Corporation

USB_HW_ClearOutPacketReady()

Description

TBD

Prototype

```
void USB_HW_ClearOutPacketReady(void);
```

USB_HW_ClrSetupEnd()

Description

TBD

Prototype

```
void USB_HW_ClrSetupEnd(void);
```

 © 2004 Micrium Technologies Corporation

3.5 OS dependent part

The OS dependent part is fairly small. It consist of a set of functions and acts as abstraction layer between the USB stack and the RTOS. If no RTOS is used, the apstraction layer will emulate the small part of the RTOS functionality required.

Routine	Explanation
USB_X_Delay()	Delays for a given number of ms.
USB_X_RestoreI()	Enable interrupts if Int-Disable-Count is 0
USB_X_DecRI()	Decrement interrupt disable count and enable interrupts if counter reaches 0
USB_X_DI()	Disable interrupts
USB_X_IncDI()	Increment interrupt disable count and disable interrupts
USB_X_SignalRx()	Wake the task waiting for receiption
USB_X_WaitRx()	Block the task until USB_X_SignalRx is called
USB_X_SignalTx()	Wake the task waiting for transmission
USB_X_WaitTx()	Block the task until USB_X_SignalTx is called

© 2004 Micrium Technologies Corporation

3.6 Generic bulk stack

The generic bulk stack is contained in the directory `USBBulk`. All "C"-files in the directory should be included in the project (compiled and linked as part of your project). The files in this directory are maintained by MICRIUM and should not require modification. All files requiring modifications have been placed in other directories.

 © 2004 Micrium Technologies Corporation

Chapter 4

The Kernel mode driver (PC)

In order to communicate with a target (client) running the stack, you have to install the USB-Bulk kernel mode driver on your Windows PC. Typically, this is done as soon as the stack runs un your target hardware.
Installation of the driver as well as how to recompile it is explained in this chapter.

© 2004 Micrium Technologies Corporation

4.1 General info about the driver

4.1.1 Why is a driver necessary ?

In Microsoft’s Windows operating systems, all communication with real hardware is implemented with kernel mode drivers. Normal applications run in "user mode". In user mode, hardware access is not permitted. All access to hardware is done through the operating system. The operating system uses a "kernel mode driver" to access the actual hardware. In other words: Every piece of hardware requires one or more kernel mode drivers to function. Windows supplies drivers for most common types of hardware, but it does not come with a generic bulk communication driver. It comes with drivers for certain classes of devices, such as keyboard, mouse and mass storage device (USB stick). This makes it possible to connect a USB mouse and not having to install a driver for it: Windows alreadys has a driver for it.
Unfortunately, there is no generic kernel mode driver which allows communication to any type of device in bulk mode. This is why a kernel mode driver needs to be supplied in order to work with the USB stack.

4.1.2 Supported platforms

The kernel mode driver works on all NT-type platforms. This includes Windows 2000 and Windows XP (home and professional). Windows NT itself does not support USB; Win98 is not supported by the driver.

 © 2004 Micrium Technologies Corporation

4.2 Installing the driver

When your target device is plugged on your computer's USB port, or when the computer is first powered up after connecting the USBBulk device, Windows will detect the new hardware.

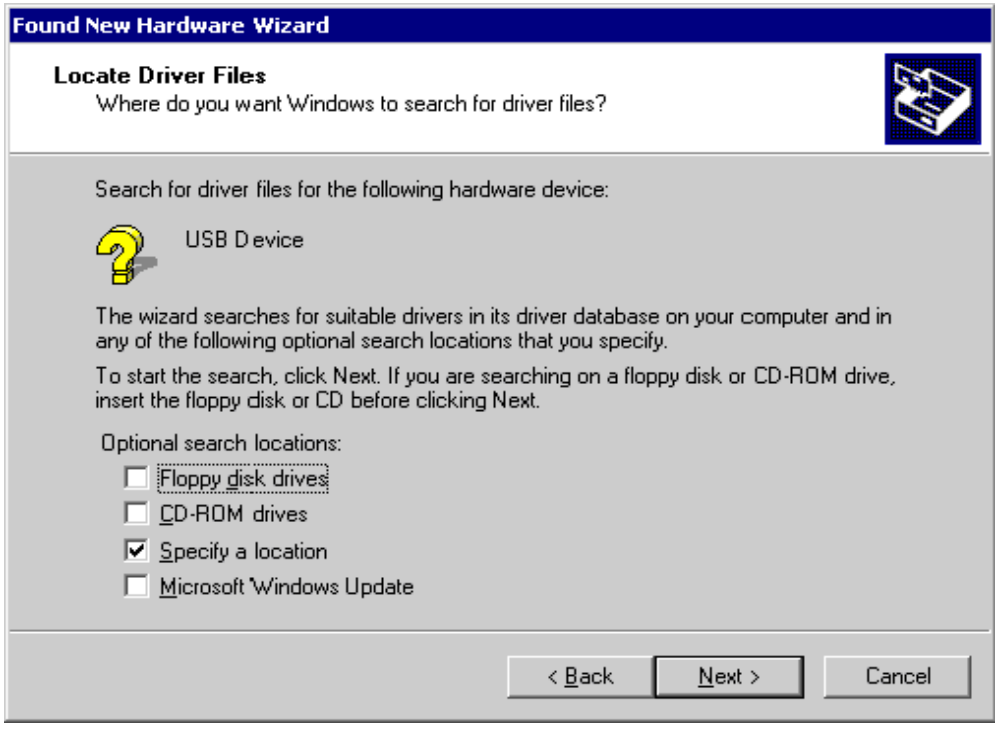
Found New Hardware Wizard
Welcome to the Found New Hardware Wizard
This wizard helps you install a device driver for a hardware device.
To continue, click Next.
< Back
Next >
Cancel

The wizard will ask you to help it find the correct driver files for the new device. First select the "Search for a suitable driver for my device (recommended)" option, then click on the "Next >" button.

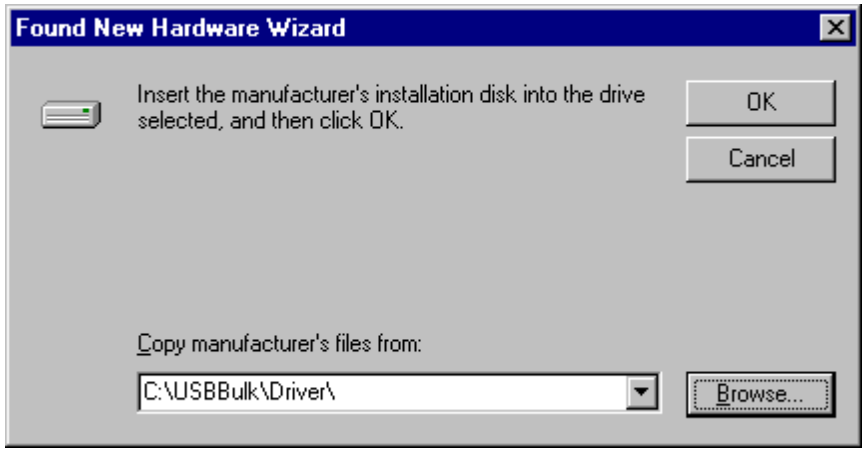
© 2004 Micrium Technologies Corporation

Found New Hardware Wizard
Install Hardware Device Drivers
A device driver is a software program that enables a hardware device to work with an operating system.
This wizard will complete the installation for this device:
USB Device
A device driver is a software program that makes a hardware device work. Windows needs driver files for your new device. To locate driver files and complete the installation click Next.
What do you want the wizard to do?
Search for a suitable driver for my device (recommended)
Display a list of the known drivers for this device so that I can choose a specific driver
< Back
Next >
Cancel

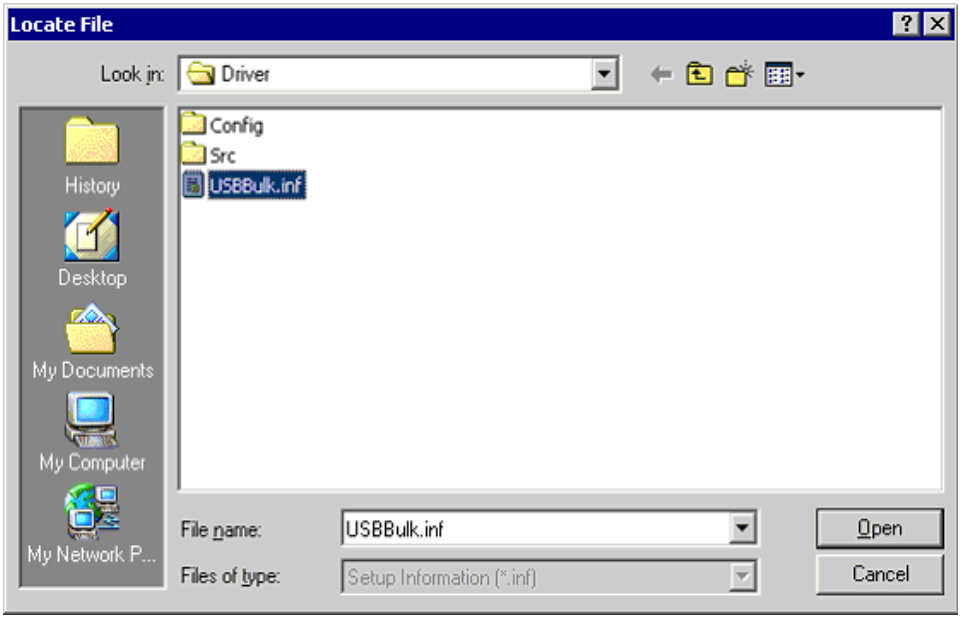
In the next step, you need to select the "Specify a location" option, and click on the "Next >" button.


Found New Hardware Wizard
Locate Driver Files
Where do you want Windows to search for driver files?
Search for driver files for the following hardware device:
USB Device
The wizard searches for suitable drivers in its driver database on your computer and in any of the following optional search locations that you specify.
To start the search, click Next. If you are searching on a floppy disk or CD-ROM drive, insert the floppy disk or CD before clicking Next.
Optional search locations:
Floppy disk drives
CD-ROM drives
Specify a location
Microsoft Windows Update
< Back
Next >
Cancel

 © 2004 Micrium Technologies Corporation



The wizard will ask you to help it find the correct driver files for the new device. Use the directory navigator to select "C:\USBBulk\DRIVER" (or your chosen location) and confirm with a click on the "Next >" button.



© 2004 Micrium Technologies Corporation

Found New Hardware Wizard
Driver Files Search Results
The wizard has finished searching for driver files for your hardware device.
The wizard found a driver for the following device:
USB Device
Windows found a driver for this device. To install the driver Windows found, click Next.
c:\usbbulk\driver\usbbulk.inf
< Back
Next >
Cancel

The wizard confirms your choice and starts to copy, when you click on the "Next >" button.

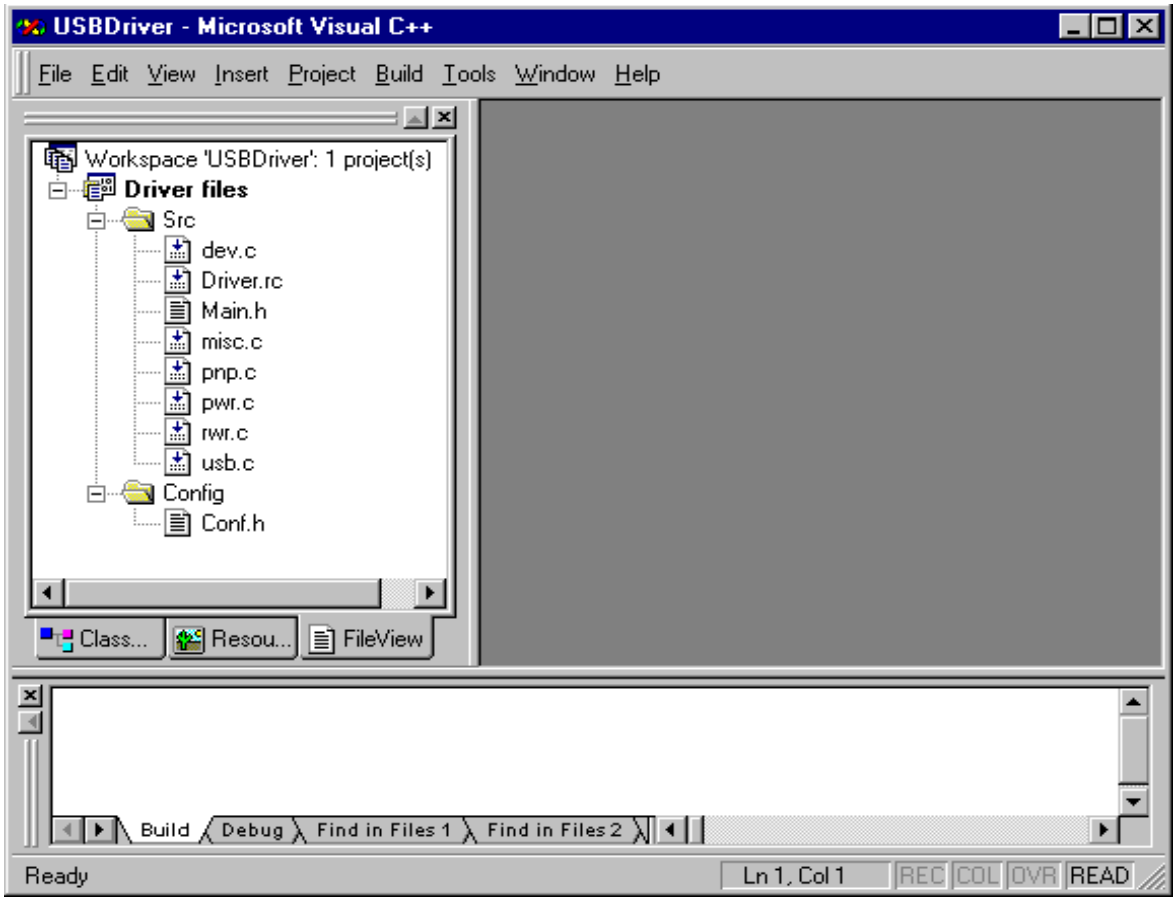
Found New Hardware Wizard
Completing the Found New Hardware Wizard
USB Bulk Device
Windows has finished installing the software for this device.
To close this wizard, click Finish.
< Back
Finish
Cancel

At this point, the installation is complete. Click on the "Finish" button to dismiss the wizard.

 © 2004 Micrium Technologies Corporation

4.3 Recompiling the driver

You need Device Developer Kit (NTDDK), as well as an installation of Microsoft Visual C6.0 or .net to recompile the driver.
The workspace is placed in the subdirectory "Driver". In order to open it, double click the workspace file USBDriver.dsw.
After double clicking it, you should see the workspace similar to the screenshot below.



Select "Build|Build USBBulk.sys" from the menu to compile and link the driver.

© 2004 Micrium Technologies Corporation

4.4 The .inf file

The .inf file is required for installation.

It looks as follows:

```
;
;
;       USB BULK Device driver inf
;
;
[Version]
Signature="$CHICAGO$"
Class=USB
ClassGUID={36FC9E60-C465-11CF-8056-444553540000}
provider=%MfgName%
DriverVer=08/07/2003

[SourceDisksNames]
1="USB BULK Installation Disk",,,

[SourceDisksFiles]
USBBulk.sys = 1

[Manufacturer]
%MfgName%=DeviceList

[DeviceList]
%USB\VID_8765&PID_1234.DeviceDesc%=USBBULK.Dev, USB\VID_8765&PID_1234

;[PreCopySection]
;HKR,,NoSetupUI,,1

[DestinationDirs]
USBBULK.Files.Ext = 10,System32\Drivers

[USBBULK.Dev]
CopyFiles=USBBULK.Files.Ext
AddReg=USBBULK.AddReg

[USBBULK.Dev.NT]
CopyFiles=USBBULK.Files.Ext
AddReg=USBBULK.AddReg

[USBBULK.Dev.NT.Services]
Addservice = USBBULK, 0x00000002, USBBULK.AddService

[USBBULK.AddService]
DisplayName    = %USBBULK.SvcDesc%
ServiceType    = 1                  ; SERVICE_KERNEL_DRIVER
StartType      = 3                  ; SERVICE_DEMAND_START
ErrorControl   = 1                  ; SERVICE_ERROR_NORMAL
ServiceBinary  = %10%\System32\Drivers\USBBULK.sys
LoadOrderGroup = Base

[USBBULK.AddReg]
HKR,,DevLoader,,*ntkern
HKR,,NTMPDriver,,USBBULK.sys

[USBBULK.Files.Ext]
```

 © 2004 Micrium Technologies Corporation

```
USBBulk.sys


;-------------------------------------------------------------------;

[Strings]
MfgName="MyCompany"
USB\VID_8765&PID_1234.DeviceDesc="USB Bulk Device"
USBBULK.SvcDesc="USB Bulk device driver"
```

red - required modifications
green - possible modifications

You have to personalise the .inf file on the red marked positions. Changes on the green marked positions are optional and not necessary for the correct function of the device.

Replace the red marked positions with your personal vendor Id (VID) and product Id (PID). These changes have to be identical with the modifications in the configuration files USB_Config.h and USB_Descriptors.c to work correct.

The required modifications of the files USB_Conf.h and USB_Descriptors.c are described in the configuration chapter.

 © 2004 Micrium Technologies Corporation

 © 2004 Micrium Technologies Corporation

Chapter 5

Using the sample application

Sample applications for both the target (client) and the PC (host) are supplied. The sample application for the PC is supplied as source code incl. MSVC6 workspace and as executable; the sample application for the target is supplied as source code only.

The sample application can be used to test the correct installation and proper function of the USB BULK device.

The PC sample application first tries to establish a connection to the target by opening the device. If this fails, an error message is shown.
If a connection can be established, it exchanges data with the target, testing the USB connection.

© 2004 Micrium Technologies Corporation

5.1 Running the sample application

The sample application is supplied in both source code and executable form in the `SampleApp` directory as `SampleApp\SampleApp.exe`. To run the sample application, simply start the executable, for example by double clicking it.
If the USB BULK device isn't connected to your PC or the driver isn't installed, you get the following message box.

If the connection to the USB BULK Device works correctly, the sample Application will write one byte and afterwards read one byte. If an error in the reading or writing process occurs a message box pops up.

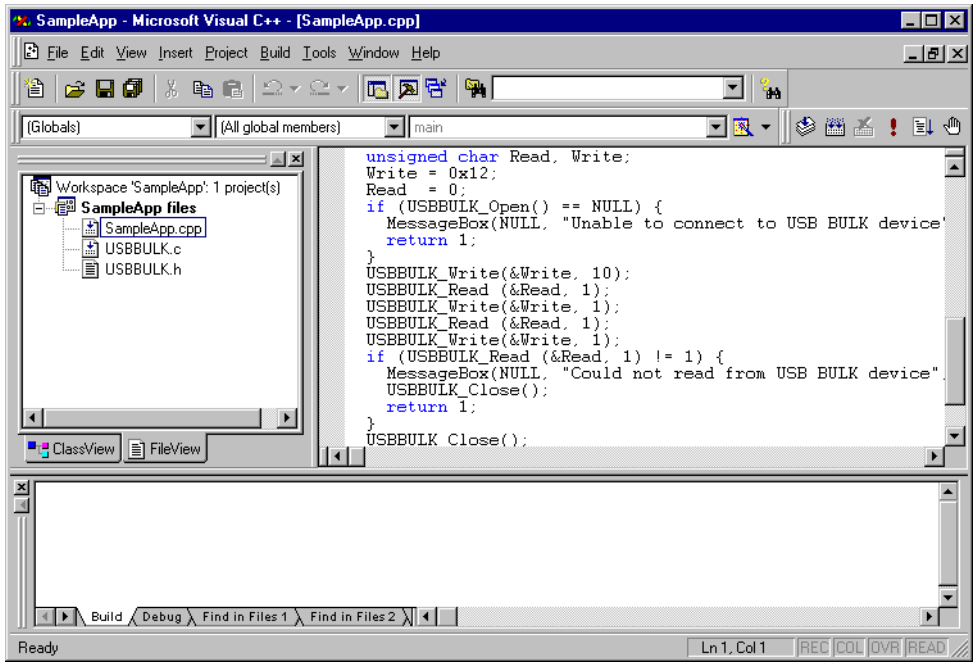
In the USB BULK device runs an application that increments the received byte and sends it back. The sample application reads this byte and checks its value. This is repeated 2 times unless an error occurs.
In either case, a message box pops up, informing you whether everything worked fine or if an error occurred.

Error Messages	Description
Unable to connect to USB BULK device	The USB device is not connected to your PC or the connection is faulty.
Could not write to device	The PC sample application was not able to write one byte.
Could not read from device (time out)	The PC sample application was not able to read one byte.
Wrong data read	The result of the target sample application isn’t correct.

 © 2004 Micrium Technologies Corporation

5.2 Compiling the PC sample application

You need for compilation of the sample application a version of a Microsoft compiler. The compiler is part of Microsoft Visual C++ 6.0 or Microsoft Visual .Net.



The source of the sample application is placed in the subfolder /SAMPLEAPP.
Open the file SampleApp.dsw und compile the source with Build|Build SampleApp.exe (Shortcut: CTRL-F7) and run the executable with Build|Execute SampleApp.exe (Shortcut: CTRL-F5) from the menu.

© 2004 Micrium Technologies Corporation

 © 2004 Micrium Technologies Corporation

Chapter 6

Configuration

Before you can use the stack in a product, you need to configure the software for your application. Configuring means modifying the files which reside in the (sub)directory `Config`.

© 2004 Micrium Technologies Corporation

6.1 Configuration files

The configuration files reside in the Config\ subdirectory of the firmware. It contains 2 files:

File	Contents
USB_Conf.h	Contains most configuration switches and some very basic USB setup options, such as Product Id and Vendor Id
USB_Descriptors.c	The USB descriptors as specified in the USB standard.

6.2 USB_Conf.h

The following table shows the available macros used for configuration:

Type	Macro	Default	Explanation
General (required) configuration			
N	USB_VENDOR_ID	0x8765	USB IF assigned vendor Id (16 bits)
N	USB_PRODUCT_ID	0x1234	arbitrary product Id (16 bits)

USB_VENDOR_ID

Description

Defines the Vendor Id of the target.

Type

Numerical

Example

```
#define USB_VENDOR_ID   0x8765
```

Additional information

The vendor Id is assigned by the USB Implementers forum (www.usb.org). For tests, you can use the default number above (or pretty much any other number). However, you may not bring a product on the market without having been assigned your own vendor Id. If you change this value, do not forget to make the same change to the .inf File as described in the Driver chapter. Otherwise the Windows host will be unable to locate the driver.

USB_PRODUCT_ID

Description

Defines the Product Id of the target.

Type

Numerical

Example

```
#define USB_Product_ID   0x1234
```

Additional information

The product Id builds in combination with the vendor Id a worldwide explicit

 © 2004 Micrium Technologies Corporation

identifier. For tests, you can use the default number above (or pretty much any other number). If you change this value, do not forget to make the same change to the .inf File as described in the Driver chapter. Otherwise the Windows host will be unable to locate the driver.

 © 2004 Micrium Technologies Corporation

6.3 USB_Descriptors.c

The USB describtors are locate in the file USB_Descriptors.c. A part of this file should be modified before using the software, other parts may be modified.

6.3.1 Manufacturer string descriptor

The descriptor is in Unicode format. The first 2 bytes are header an should not be changed. Changes in this string do not affect the functionality at all.

```
const unsigned char USB_acManfString[20] = {
  sizeof(USB_acManfString),
  USB_DESC_TYPE_STRING,
  'M',0,
  'Y',0,
  'C',0,
  'O',0,
  'M',0,
  'P',0,
  'A',0,
  'N',0,
  'Y',0,
};
```

6.3.2 Product string

The descriptor is in Unicode format. The first 2 bytes are header an should not be changed. Changes in this string do not affect the functionality at all.

```
const unsigned char USB_acProdString[32] = {
  sizeof(USB_acProdString),
  USB_DESC_TYPE_STRING,
  'U',0,
  'S',0,
  'B',0,
  '-',0,
  'B',0,
  'U',0,
  'L',0,
  'K',0,
  ' ',0,
  'D',0,
  'e',0,
  'v',0,
  'i',0,
  'c',0,
  'e',0,
};
```

 © 2004 Micrium Technologies Corporation

Chapter 7

Support

This chapter should help if any problem occurs. This could be a problem with the tool chain, with the hardware, the use of the USB functions or with the performance and it describes how to contact the support.

© 2002-2004 Micrium Technologies Corporation

7.1 Problems with tool chain (compiler, linker)

The following shows some of the problems that can occur with the use of your tool chain. The chapter tries to show what to do in case of a problem and how to contact the support if needed.

7.1.1 Compiler crash

You ran into a tool chain (compiler) problem, not a problem with the software. If one of the tools of your tool chain crashes, you should contact your compiler support:

```
"Tool internal error, please contact support"
```

7.1.2 Compiler warnings

The code of the software has been tested with different compilers. We spend a lot of time on improving the quality of the code and we do our best to avoid compiler warnings. But the sensitivity of each compiler regarding warnings is different. So we can not avoid compiler warnings for unknown tools.

Warnings you should not see

This kind of warnings should not occur:

```
"Function has no prototype"
"Incompatible pointer types"
"Variable used without having been initialized"
'Illegal redefinition of macro'
```

Warnings you may see

Warnings such as the ones below should be ignored:

```
"Integer conversion, may loose significant bits"
'Statement not reached"
"Meaningless statements were deleted during optimization"
```

Most compilers offers a way to supress selected warnings.

7.1.3 Compiler errors

We assume that the used compiler is ANSI 'C' compatible. If it is compatible there should be no problem to translate the code.

7.1.4 Linker problems

Undefined externals

If your linker shows the error message "Undefined external symbols..." please check if all required files have been included to the project.

7.2 Problems with hardware/driver

If your tools are working fine but your display does not work may one of the following helps to find the problem.

Stack size to low?

Please make sure there have been configured enough stack. We can not estimate exactly how much stack will be used by your configuration and with your compiler.

 © 2002-2004 Micrium Technologies Corporation

7.3 Contacting support

If you need to contact the support, please send the following information to support@micrium..com:

- A detailed description of the problem
- The configuration file `USB_Conf.h`
- The configuration file `USB_Descriptor.c`
- The error messages of the compiler

7.4 FAQ's

Q: Which CPUs can I use the software with?
A: It can be used with any CPU (or MPU) for which a "C" compiler exists. Of course, it will work faster on 16/32-bit CPUs than on 8-bit CPUs.

© 2002-2004 Micrium Technologies Corporation

 © 2002-2004 Micrium Technologies Corporation

Index

Symbols
"C" compiler 8
"C" programming language 8

A
ANSI 8

C
Config folder 39
Configuration, of uC/USB 39–42

D
Data types used 9

S
Support 43–45
Syntax, conventions used 9

U
Using the sample application 35–37

© 2004 Micrium Technologies Corporation